Emergent socially rational Utilitarianism in a Peer-to-Peer System

Overview

University of Bologna - Dept. of Computer Science - Andrea Marcozzi and David Hales

Abstract

For many applications peer-to-peer (P2P) systems require their member nodes (or agents) to behave in a socially beneficial way. This requirement is known as the “Principle of Social Rationality” (Kalenica 1999): if an agent has a choice of actions it should chose the action that maximizes the social utility. This principle can be contrasted with classical individual rationality that states agents should select actions that maximize their individual utility.

Recently, simple locally adaptive protocols have been proposed (Marcozzi 2005, Hales 2005) that claim to produce socially rational outcomes through a process of self-organization even though nodes only act on their own utility values. We took the SkillWorld model (Hales 2005) and modify the utilities to explore a large space of possible values. In each case we checked if the protocol maximized the collective utility or not. This new model is called “ResourceWorld”.

The ResourceWorld Model

The model represents the situation in which nodes in a peer-to-peer network can store and serve some resources (R). Each node may have a maximum of 20 links to other nodes (peers). Each link is bidirectional.

![Diagram](Image)

Fig. 1. An illustration of the “ResourceWorld” model. Shading of nodes represents strategy, the number inside the node indicates the resource in (a) node i receives a certain request j, since it has the appropriate resource to satisfy it, it gains the benefit payoff (b=2); in (b), i don’t hold the required resource, since it is an altruistic node, it can pass it to its neighbor j, which has the right resource; in case (c), it gains the b payoff and i pays the c payoff (c=20). In (c), since node i is a selfish node, it can not pass the job to its neighbor.

Experiment configuration and results

We performed a massive number of experiments with this model; we played with the utilities (see figure 3) with the aim to explore a large space of possible values. Both benefit (b) and cost (c) payoffs were ranging from 0.1 to 2.0 by steps of 0.1. For each configuration we performed 10 different runs and we took the average of the results. Hence we performed 20 * 20 * 10 = 4000 different runs.

The utility measure we adopted is the percentage of Completed Jobs (Pcj), which is the average between the number of request submitted and the number of request completed at each cycle. Figure 2 shows the number of cycles needed to obtain a good level of Pcj (by good Pcj we mean a value greater than 80%) fixing the benefit payoff (b=1) and varying the cost (from 0.1 to 1). From figure 3 we found that to obtain a good Pcj, the benefit payoff must be greater or equal than than the cost payoff (b ≥ c). When b < c we obtain a very low Pcj (ranging from 25% to 35%); when b = c, the system will take longer to achieve this. To obtain a good level of Pcj in a small time, c must be smaller than the half of b (c < 1/2b).

Future Works

Varying the benefit and cost payoff we obtained two interesting rules (see table 2). It seems that under these rules the system gives good outcomes. We think that these results may be influenced by the topology of the network and by the number of skills. What happens if we perform the same experiments with networks having a small degree? Or what happens, if the number of skill involved is smaller or bigger then 20? This might be the subject of future research.

Table 1: Nodes state

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altruism flag</td>
<td>A ∈ {0, 1}</td>
</tr>
<tr>
<td>Resource/Skill type</td>
<td>R ∈ {1, 2, 3, 4, 5}</td>
</tr>
<tr>
<td>Maximum view size</td>
<td>d = 20</td>
</tr>
<tr>
<td>Utility</td>
<td>U ∈ R</td>
</tr>
</tbody>
</table>

Table 2: Rules discovered from the experiments

<table>
<thead>
<tr>
<th>Time to 80% Pcj</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>181...740 cycles</td>
<td>c ≤ b</td>
</tr>
<tr>
<td>90...180 cycles</td>
<td>c &lt; 1/2b</td>
</tr>
</tbody>
</table>

Contact

Andrea Marcozzi
Dept. of Computer Science
Mura Anteo Zamboni 7
40127 Bologna, Italy
Fax: +39 051 2094510
Phone: +39 051 2094515
E-Mail: marcozzi@cs.unibo.it
URL: www.cs.unibo.it/~marcozzi

David Hales
Dept. of Computer Science
Mura Anteo Zamboni 7
40127 Bologna, Italy
Fax: +39 051 2094510
Phone: +39 051 19880461
E-Mail: dav@davidhales.com
URL: www.davidhales.com

DELIIS: Integrated European Project founded by the "Complex Systems" Proactive Initiative within the Sixth Framework Programme